Python

Modules and Packages

@kabirbaidhya

Reflections

So far,

We've covered all these things

1. Python Basics, Conditionals and Loops
2. Functions and Exception Handling

Note: If you're not aware of these. Read them at https://github.com/kabirbaidhya/learn-python-django-web

Modules

What are Modules?

Modules in Python are nothing just plain python files with `.py` extension that may contain and expose a set of definitions be it functions, classes, variables, etc.

Definitions from one module can be imported to other modules and so on which helps in code reusability.

You can use `import` statement to import a module or the definitions from a module in another module.

What modules offer?

- Namespacing - avoid naming colisions of definitions
- Portability
- Even better code reusability
- Better Code Organization
- Modularity - Of Course

Example 1

This is a file (module) named `test.py`.

```
def factorial(n):
    if n == 0:
        return 1

    return n * factorial(n - 1)


def fibo(n):
    result = []
    a, b = 0, 1
    while b < n:
        result.append(b)
        a, b = b, a+b

    return result
```

Example 1

This is another file (module) `main.py` that uses functions from `test` module.

```
import test # Import the test module here


def main():
    n = input('Enter a Number: ')

    series = test.fibo(n).join(', ')
    print('Series upto {}: \n {}'.format(n, series))
    print('Factorial of {}'.format(n, test.factorial(n)))


main()
```

More on Modules

You application generally is composed of several different modules. Modules may contain both executable and non-executable code (like function definitions, class definitions etc).

But it's usually preferable to keep all your executable code only in your main module or the module that you're using as an entry point script and make all the other modules have only non-executable code.

Module's `__name__`

In every module you define you can find a `__name__` variable whose value would be the name of the module.

When you run a module directly as a script, the value of `__name__` inside that module is set to `__main__` otherwise it's value would be the module's name.

Example 2

```
# fib.py
import sys

def fib(n):
    a, b = 0, 1
    while b < n:
        print(b, end=' ')
        a, b = b, a+b
    print()


if __name__ == '__main__':
    n = int(sys.argv[1])
    fib(n)
```

You can use this file both as a script and an importable module.

Example 2 - Running it directly

Run the script from command line

```
$ python fib.py 100
```

This would take the value of `n` from the command line arguments and run the program. The code executes and prints the output since here the value of `__name__` will be `__main__`.

Example 2 - Importing as a module

```
import fib

n = int(input('Enter N: '))
# Now execute that function.
fib.fib(n)
```

Now if you're importing the module like this, the executable code under the `if __name__ == '__main__'` block won't be executed.

Instead it will be executed when the function `fib` is called.

The `import` Syntax Variants

Syntax 1

rts the module itself in it's own name.

```
import fib
```

So, to access the `fib` function defined inside the `fib` module you need to do

```
>>> fib.fib(50)
```

Syntax 2

Imports the function `fib` from the module `fib`.

```
from fib import fib
```

Now to call the `fib` function you can directly do

```
>>> fib(50)
```

You can also import multiple definitions from a module.

```
from foo import bar, baz, test, xyz
```

Syntax 3

Imports all the names that the module defines

```
from foo import *
```

Note: This is not considered a good practice and not recommended either.

Built-in Modules

Python provides it's standard library as a set of modules and packages. Let's look into this simple example.

There are many built-in modules & packages available out of the box in python's standard library. You can check the full list here.

Example 3

Using python's built-in `math` module

```
import math

value = 5.34

print('value =', value)
print('ceil =', math.ceil(value))
print('floor =', math.floor(value))
print('abs =', math.fabs(value))
```

Here we're using the `math` module which is one of the built-in modules that python has in it's standard library.

Packages

Packages in Python

If you think of the modules as files having `.py` extension; then you can think of packages as directories that contains modules.

Most importantly to be a package the directory, must contain a special file `__init__.py` which indicates python that it's a package that contains modules.

Example 4

For instance, consider the following directory structure:

```
foo/
  __init__.py
  bar
    __init__.py
    baz.py
```

Here `foo` is the main package, `bar` is the sub-package under `foo` and `baz.py` is a module inside `bar`.

Example 4

Let's say `baz.py` has the following code:

```
def say_hello(to = 'World'):
    print('Hello', to)
```

Example 4

Now to reference the `baz` module from outside you would use the `import` statement with dotted names like this:

```
import foo.bar.baz

# You can call the say_hello() using
foo.bar.baz.say_hello()
```

Alternatively you can use the `import .. from` syntax like this:

```
form foo.bar.baz import say_hello

say_hello('Again')
```

Let's do some real coding :)

Building a Simple App

User Information App

Phase I

Please follow this link for the steps.

https://github.com/kabirbaidhya/learn-python-django-web/blob/master/units/python/7/modules-and-packages.md

Read More?

Links

Want to read more? Go through these links.

1. https://docs.python.org/3/tutorial/modules.html
2. https://python.swaroopch.com/modules.html
3. https://docs.python.org/3/library/
4. https://docs.python.org/3/library/json.html#module-json
5. https://docs.python.org/3/library/functions.html#open

This slide was a part of course

Python, Django & Web Development

github.com/kabirbaidhya/learn-python-django-web

Thank You

@kabirbaidhya

kabirbaidhya@gmail.com